VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro podporu vícekriteriální hodnocení variant

Decision Supporting System for Multicriterial Evaluation of Variants

2010 Roman Zajíc

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

datum: 7. 5. 2010 podpis: ……

Abstrakt

Tato bakalářská práce se zabývá oblastí lidského rozhodování, konkrétně se zaměřuje na možnosti usnadnění tohoto procesu v případě náročnějších rozhodnutí. Lidské rozhodování je ve své podstatě velmi komplikovaný proces a proto lidé mají často tendence obtížnější rozhodnutí odkládat, či delegovat na jiné osoby, což nemusí být vždy správná volba. Náplní této práce je vývoj aplikace pro zjednodušení a zefektivnění procesu rozhodování využitím metody multikriteriálního hodnocení variant. Tato metoda umožňuje rozložit řešený problém na několik dílčích součástí, rozhodnout jednotlivě o těchto částech, matematicky zpracovat získaná data a následně využít získané výsledky pro finální rozhodnutí. Umět se v dnešní době správně a efektivně rozhodovat je, dle mého názoru, ceněná vlastnost a tato aplikace může pomoci jak jednodušeji realizovat rozhodnutí, tak lépe pochopit podstatu problému a učinit tak nejlepší možné rozhodnutí.

Klíčová slova: Java, XML, Multikriteriální hodnocení variant

Abstract

This bachelor thesis deals with the human decision-making process, specifically focusing on ways to make the process easier for challenging decisions. Human decision making is inherently a very complicated process and therefore people have a tendency to often delay difficult decisions or delegate to others, which may not always be the right choice. The content of this work is to develop application for simplifying and streamlining the decision-making process by using the method of multicriterial evaluation of variants. This method allows you to split the solved problem into several sub-components, to decide individually on those parts, mathematically process the acquired data and then use the obtained results for the final decision. The ability to decide correctly and efficiently is today, in my opinion, very prized characteristics and this application can help both easier to implement decisions or better understand the nature of the problem and make the best decision.

Key words: Java, XML, Multicriterial evaluation of variants

Obsah

1. Úvod

V úvodní části bych rád osvětlil své důvody pro výběr právě této bakalářské práce a uvedl, jakým směrem jsem se vydal pro úspěšnou realizaci výsledné aplikace. Téma Multikriteriální hodnocení variant jsem si zvolil ze dvou hlavních důvodů.

Prvních z nich je problematika rozhodování sama o sobě, protože ať chceme nebo nechceme, člověk se musí rozhodovat každý den o více či méně náročných věcech. Nebudu zde popisovat, jakým způsobem probíhá samotný proces lidského rozhodování, ale zaměřím se právě na oblasti, kde má smysl uvažovat o zjednodušení a zefektivnění. Pokud člověk stojí před důležitým rozhodnutím mezi několika variantami, které mají složitějších charakteristiku, pak má smysl o tomto rozhodnutí více uvažovat. Jednou z metod řešení je definování kritérií, která jsou pro nás ta hlavní. Na základě určení jejich mezi sebou a ve vztahu k jednotlivým variantám, lze díky matematickým výpočtům dospět k jednoznačnému výsledku.

Druhým důvodem byla již vytvořená aplikace IVMULTI sama o sobě. Tato aplikace byla vytvořena již roku 1990 a od té doby má stále své uplatnění. To znamená, že smysl celé aplikace stále zůstává nezměněn a je i nadále využíván v praxi. Za dobu 20 let však informační technologie urazili hodný krok kupředu a i přesto, že aplikace je stále funkční, tak již neodpovídá moderním požadavkům a působí zastarale. Myšlenka vytvořit novou aplikaci, která by zachovala původní funkcionalitu a implementovala navíc nové možnosti, mi přišla zajímavá.

Na základě těchto dvou důvodů jsem zažádal o možnost zpracovat bakalářskou práci na toto téma a po odsouhlasení jsem na celé aplikaci začal pracovat. V této bakalářské práci bude dále uveden průběh od jednotlivých fází analýz obou aplikací přes postup implementace až po testování, vytvoření dokumentace a závěrečné zhodnocení.

2. Zadání práce

Ze zadání této bakalářské práce je možné vyčíst několik podstatných požadavků na charakteristiku nové aplikace:

- Vytvoření moderní aplikace odpovídající dnešním standardům
- Plně grafické uživatelské prostředí
- Zachování funkcionality původní aplikace

Dodatečně byly doplněny také další požadavky ze strany zadávajícího:

- Multiplatformní aplikace
- Neomezený počet zadaných kritérií a variant
- Export dat bude řešen pomocí XML souborů

Stanoven byl také základní postup pro úspěšné dokončení jak aplikace, tak samotné práce.

1. Analýza původní aplikace IVMULTI

2. Analýza nové aplikace později nazvané MULTI
3. Implementace
4. Testování a ladění

V následujících kapitolách bude vždy dopodrobna rozepsán postup jednotlivých fází, dosažené výsledky a závěrečné zhodnocení.

3. Analýza stávající aplikace IVMULTI

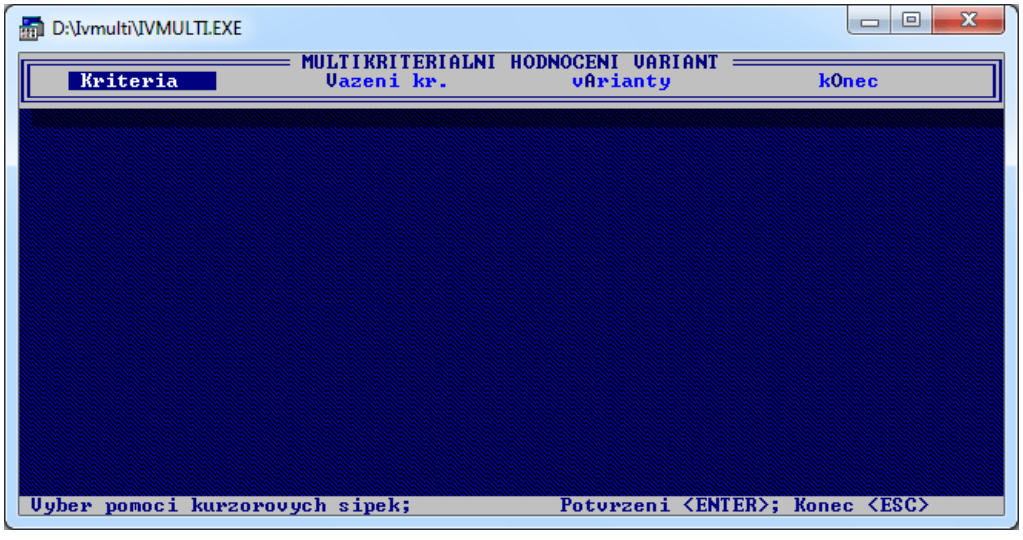
3.1 Teoretický základ

Aplikace má umožnit zefektivnění procesu rozhodování v oblasti řízení využitím metody multikriteriálního hodnocení variant. Teoretický základ této metody byl je přehledně popsán v Uživatelském manuálu systému IVMULTI, který je k dispozici jako příloha této práce a proto se zde nebudu zabývat detailnějším rozepsáním dané problematiky.

3.2 Základní práce s aplikací

Aplikace IVMULTI byla vytvořena Ing. Ivo Vondrákem, CSc. roku 1990. Program byl napsán v jazyce C a primárně byl určen pro počítače s operačním systémem MS-DOS. Implementováno bylo kompletní grafické rozhraní, které bylo ovládáno za pomocí klávesnice. K testování a analýze samotné byl poskytnut kompletní zdrojový kód, včetně uživatelské dokumentace a program samotný obsahující testovací soubory. V následujících několika odstavcích bude proto nyní popsána základní práce s aplikací.

Vše se spouští velmi jednoduše souborem IVMULTI.exe. Po vypsání úvodní obrazovky se jménem aplikace, autorem a autorských právech se spustí základní obrazovka, která bude dostupná během celého běhu aplikace.



Obr. 1: IVMULTI – Hlavní obrazovka

Na Obr. 1 můžeme vidět strukturu hlavní obrazovky. Menu je jednoduše rozděleno na 4 základní části Kritéria, Vážení Kritérií, Varianty a Konec. Práce s celou aplikací probíhá logicky z levé strany na pravou. Ve stavovém řádku je vždy vypsána krátká nápověda pro aktuální kroky v rámci aplikace.

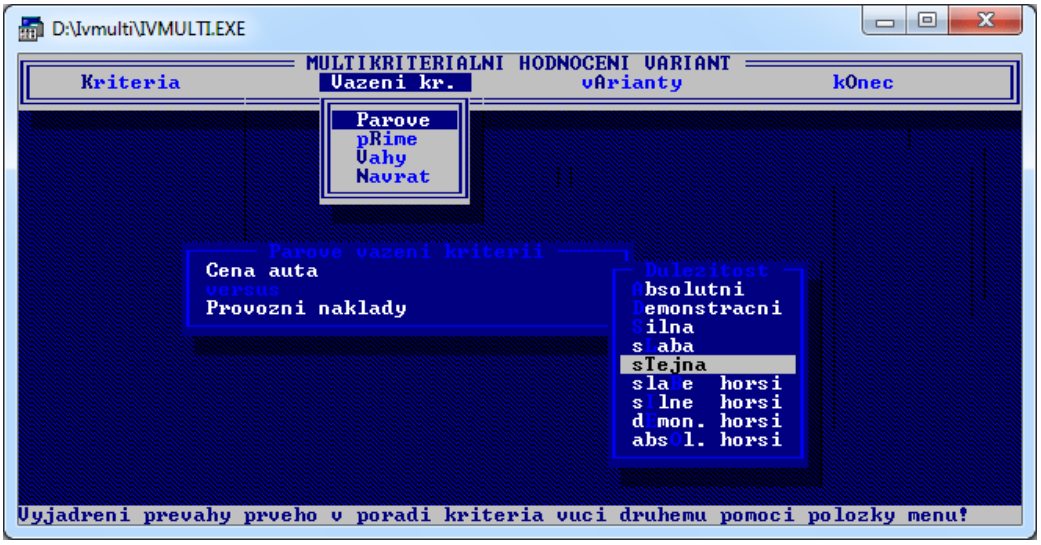
Po spuštění aplikace má uživatel 2 možnosti. První z nich je zadání souboru, kde již dříve byla uložena potřebná data pro výpočet. Nutno podotknout, že mohou být uložena a tudíž také načtena pouze data týkající se názvu a vah kritérií, nikoli variant. Pro načtení souboru existuje následující rozhraní, které vypíše obsah adresáře, odkud je aplikace spuštěna (Obr. 2)

Obr 2 : UVMULTI - Načtení souboru

Pokud uživatel načte již existující soubor s daty, bude automaticky vložen počet požadovaných kritérií s názvy, případně také s váhami pokud jsou obsaženy v souboru. Jestli se uživatel rozhodne vytvořit nový projekt, musí ručně zadat požadovaný počet kritérií, který je limitovaný počtem 10. Následuje zadání názvu kritérií, které je možné kdykoli změnit (Obr. 3).

Obr. 3: IVMULTI - Editace názvu kritérií

Pokud byly správně zadány názvy všech kritérií, je možné přistoupit k dalšímu kroku a to párovému vážení kritérií (Obr. 4). Tento krok postupně projde všechny párové dvojice a umožní stanovit jejich důležitost mezi sebou. Využívá k tomu předem definovanou škálu tzv. „Fuzzy hodnot“, kterou můžeme vidět na následujícím obrázku. Na tomto konkrétním příkladu máme vybrána kritéria „Cena auta“ a „Provozní náklady“, zároveň je vybrána důležitost „Stejna“ což pro nás znamená, že váha obou kriterií je pro nás stejně důležitá. Pokud by pro nás bylo nejdůležitější kritérium „Cena auta“ zvolili bychom důležitost „Absolutni“. V opačném případě volbu „Absol. horsi".



Obr. 4: IVMULTI – Párové vážení kritérií

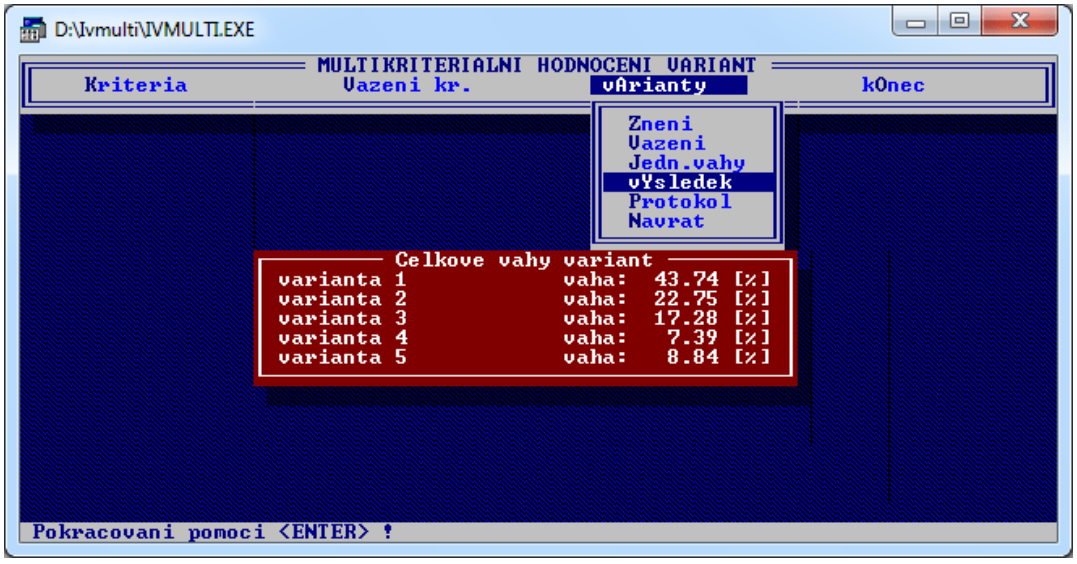
Po zadání párového vážení všech dostupných kritérií proběhne kontrola a automatická oprava zadaných důležitostí. Po úspěšném zadání všech vah je možné si tyto váhy nechat vypsat, či je také přímo editovat. I tato editace je ošetřena, aby nedošlo k zadání nepřípustných hodnot. Jakmile máme zadána všechna kritéria, můžeme přistoupit k vytvoření množiny variant. Postup je obdobný jako při tvorbě kritérií tzn., stanovíme počet variant (opět limitovaný počtem 10), zadáme jejich názvy a přistoupíme k jejich párovému vážení. Ukázka na Obr. 5.

Obr 5: IVMULTI – Párové vážení variant

Postup párového vážení probíhá opět obdobně jako u kritérií s tím rozdílem, že vždy proběhne postupné vážení všech variant pro jedno z kritérií. Následuje opět automatická kontrola vah jednotlivých variant a jejich případná změna. Po skončení vážení je možné si všechny váhy nechat vypsat ovšem není zde možné je přímo změnit (Obr. 6.).

Obr. 6: IVMULTI – Zobrazení variant

Jestliže máme zadány všechny potřebné kritéria, varianty a jejich váhy tak můžeme přistoupit k finální části a to zpracování výsledku ze zadaných dat. Jako výsledek dostaneme výpis všech variant a jejich procentuální váhy (Obr. 7). Aplikace zahrnuje i možnost tisku vypočteného výsledku, ovšem na svém počítači se systémem Windows 7 se mi tuto funkci nepodařilo spustit.



Obr. 7: IVMULTI – Výsledek

3.3 Struktura aplikace

Jak jsem již dříve uvedl, aplikace byla vytvořena v jazyce C bez použití objektového přístupu. Skládá se proto z jednoho zdrojového souboru IVMULTI.C. Celý soubor je přehledně rozčleněn na deklaraci proměnných a deklaraci všech funkcí.

Hlavní proměnné jsou definovány jako struktury obsahující pole zadaných názvů a vah pro kritéria i pro varianty. Také je zde vytvořeno pole tzv. „fuzzy hodnot“ potřebné pro párové vážení. S těmito proměnnými pak pracují všechny funkce. Jako příklad zde uvedu deklaraci struktury pro varianty.

```
struct elem2 {
    char  t[21];
    float v[10];
};

struct elem2 varianta[] = {
    {"varianta 1",0,0,0,0,0,0,0,0,0,0}, /* Text variant a jejich vahy */
    {"varianta 2",0,0,0,0,0,0,0,0,0,0},
    {"varianta 3",0,0,0,0,0,0,0,0,0,0},
    {"varianta 4",0,0,0,0,0,0,0,0,0,0},
    {"varianta 5",0,0,0,0,0,0,0,0,0,0},
    {"varianta 6",0,0,0,0,0,0,0,0,0,0},
    {"varianta 7",0,0,0,0,0,0,0,0,0,0},
    {"varianta 8",0,0,0,0,0,0,0,0,0,0},
    {"varianta 9",0,0,0,0,0,0,0,0,0,0},
    {"varianta 10",0,0,0,0,0,0,0,0,0,0}
```

Jednotlivé funkce jsem si poté logicky rozdělil na dvě hlavní kategorie:

1. funkce zajišťující grafické rozhranní a práci s hlavními proměnnými
2. funkce realizující výpočty nezbytné pro požadované multikriteriální hodnocení

ad 1. funkce první kategorie:

```
void uloz_soub(){…},int  cti_dir(char *maska,char *soubor){…}, void
nacti_soub(void){…}, void editkrit(void){…}, void nova_krit(void){…}, a další…
```

Funkce obsažené v této kategorii realizují kompletní tvorbu grafického rozhraní celé aplikaci a umožňují uživateli interaktivně pracovat s programem. Realizují načtení souborů s daty, zápis a editaci kritérií a také vah. Zařadil jsem zde i funkci Main, která realizuje vytvoření základní struktury aplikace s Menu a samozřejmě běh celé aplikace.

ad 2. funkce druhé kategorie:

```
void par_krit(void){…}, void vl_vek(int n,float eps,float a[][10],float v[],char
t){…}, void uzaver(int rad,float a[][10],char t){…}, a další
```

Tyto funkce umožňují výpočetní jádro celé aplikace. Mají za úkol zpracovat uživatelem zadané důležitosti na hodnoty požadované pro výpočet. Provést opravu pokud se zadané hodnoty nacházejí mimo požadovanou škálu a na závěr provést finální výpočet.

3.4 Shrnutí a závěr

Díky této analýze jsem dospěl k několika klíčovým závěrům, potřebným pro realizaci nové aplikace.

1. **Zachování hlavní myšlenky:**

   Hlavní myšlenka běhu celé aplikace sestávající z fází vytvoření kritérií, párového vážení, vytvoření variant, párového vážení a výpočtu výsledku by měla zůstat zachována, jelikož tento postup je potřebný pro správný výpočet výsledků.

2. **Nové grafické rozhraní:**

   Grafické rozhraní a funkce, které jej vytvářejí, budou muset být všechny nově implementovány, protože není možný jejich přenos na modernější platformu. Také design aplikace již neodpovídá aktuálním standardům.

3. **Přenesení již implementovaných funkcí**

   Funkce realizující hlavní výpočty pro kontrolu mezí párových vah, vlastního vektoru, tranzitivního uzávěru a samotných výsledků budou moci být zachovány při drobných změnách kódu z jazyka C do jazyka Java.

4. **Import a Export**

   Funkce umožňující import a export dat budou také zachovány, ovšem nově implementovány. Bude rovněž zvolen novější formát souborů.

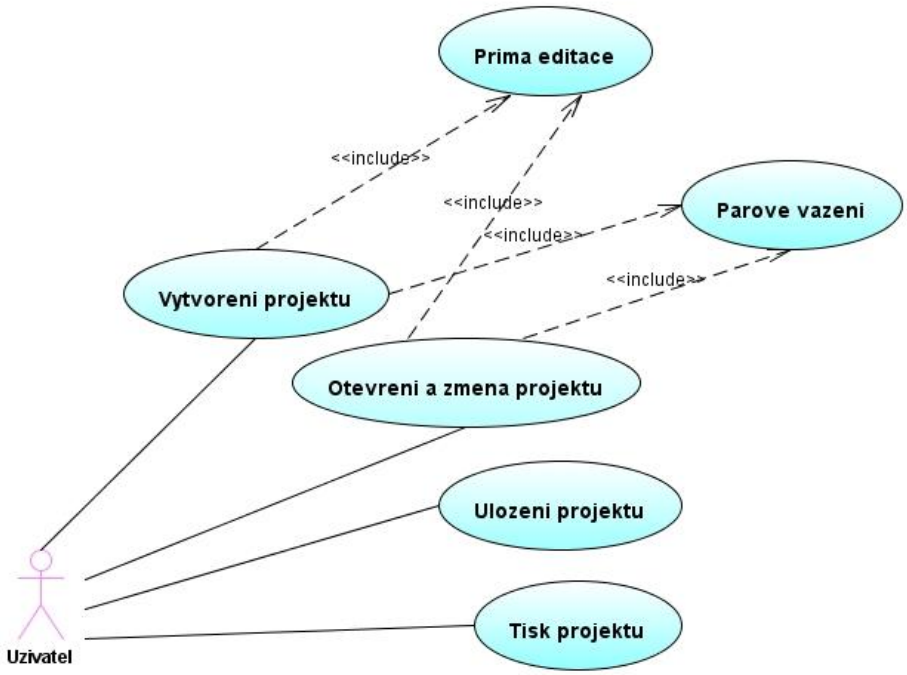
4. Analýza nové aplikace MULTI

4.1 Objektová analýza

Cílem objektové analýzy bylo nastínit základní chování aplikace, které bylo následně využito při vytváření grafického návrhu a samotné implementaci. Jako pomocný nástroj pro celou objektovou analýzu jsem zvolil modelovací jazyk UML. Grafy byly vytvořeny v programu Microsoft Visio 2007.

4.1.1 Případy užití (Use Case)

První krok v rámci analýzy sestával z vytvoření diagramu případů užití, tzv. Use Case diagram. Tento diagram názorně zobrazuje základní práci uživatele (aktéra) s aplikací. Jelikož se jedná o poměrně jednouchou aplikaci, byl vytvořen pouze jeden hlavní diagram. Ostatní důležité případy užití byly popsány slovně.



Obr. 8 – Use Case diagram

Případ užití: Vytvoření projektu

1. Uživatel zadá volbu „Nový projekt“
2. Systém zobrazí formulář pro vytvoření projektu
3. Uživatel zadá název projektu a požadovaný počet kritérií
4. Systém zobrazí formulář pro zadání kritérií a nabídne 2 možnosti určení jejich vah (přímo/párově)
5. Uživatel zadá kritéria a určí jejich váhy
6. Systém zobrazí formulář pro zadání variant a opět nabídne 2 možnosti určení jejich vah (přímo/párově)
7. Uživatel zadá varianty a určí jejích váhu
8. Systém zobrazí přehled všech zadaných informací, nabídne možnost jejich změny (přímo/párově) a vypočte výsledek multikriteriálního hodnocení variant na základě zadaných informací

Případ užití: Párové vážení

1. Uživatel zvolí možnost „Párové vážení“
2. Systém zobrazí formulář pro párové vážení zvolených kriterií/variant podle stanoveného pořadí
3. Uživatel vybere za pomoci posuvníku požadovanou důležitost
4. Systém a uživatel opakují kroky 2 a 3 dokud nejsou stanoveny důležitosti pro všechny kritéria/varianty
5. Systém zobrazí vypočtené váhy získané po zadání důležitostí

Případ užití: Přímá editace

1. Uživatel zvolí možnost „Přímá editace“
2. Systém zpřístupní tabulky obsahující názvy kritérií/variant a jejich váhy pro přímou editaci
3. Uživatel změní požadované údaje
4. Systém zkontroluje změnu údajů a opět je uzamkne

Případ užití: Uložení projektu

1. Uživatel zadá volbu „Uložit projekt“
2. Systém zobrazí formulář pro specifikaci a uložení dat ve formě XML souboru
3. Uživatel vybere požadovaná umístění a potvrdí uložení
4. Systém uloží data a zobrazí potvrzení o úspěšném uložení

Případ užití: Otevření projektu

1. Uživatel zadá volbu „Otevřít projekt“
2. Systém zobrazí formulář pro vyhledání XML souboru s daty
3. Uživatel vybere požadovaný soubor
4. Systém zobrazí všechny uložené informace a vypočte výsledek multikriteriálního hodnocení variant

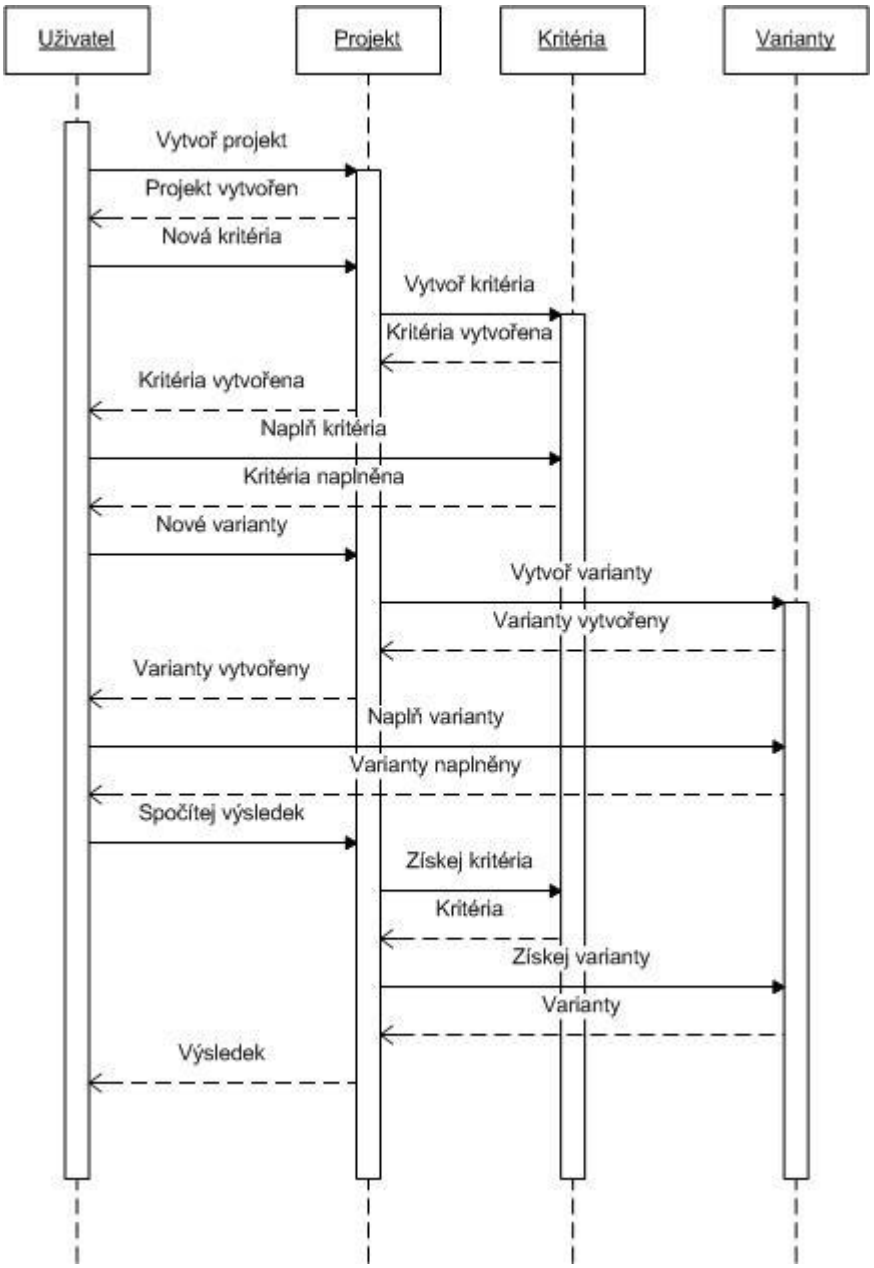
Případ užití: Tisk projektu

1. Uživatel zadá tisk projektu
2. Systém zobrazí dialog pro nastavení tisku
3. Uživatel nastaví a potvrdí tisk
4. Systém vytiskne projekt a zobrazí potvrzení o úspěšném tisku

4.1.2 Sekvenční diagram

Sekvenční diagramy znázorňují životní cyklus aplikace. Jsou zde zobrazeny instance jednotlivých tříd (zobrazeny jako obdélníky). Komunikace mezi těmito objekty je vyjádřena vodorovnými šipkami. Šipka tvořena plnou čárou znázorňuje zprávu odesílanou z objektu A na objekt B, čárovaná šipka pak značí zpětnou odpověď.

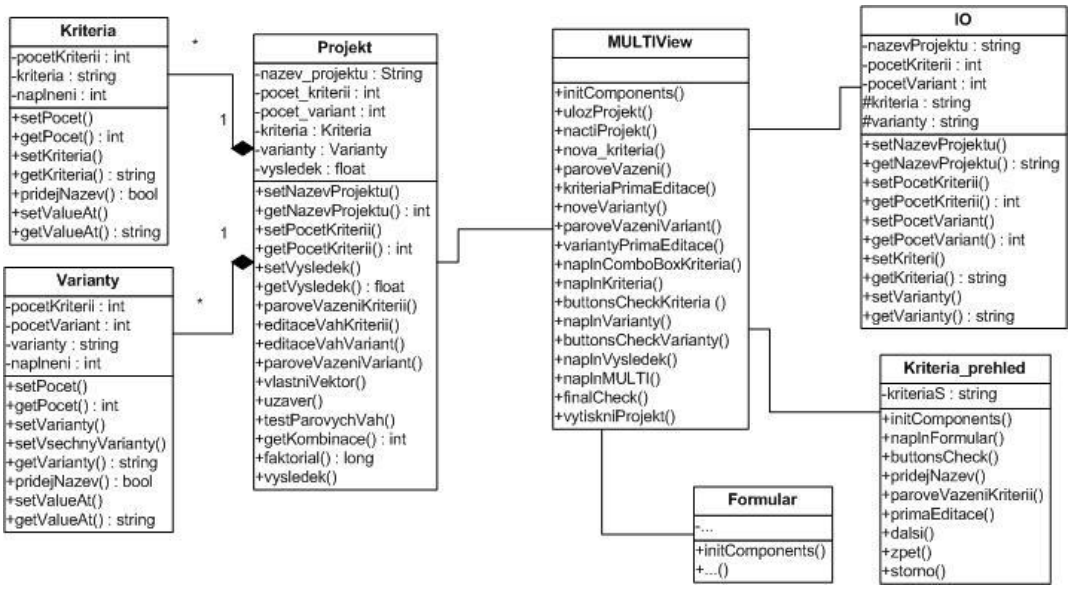
Jako příklad sekvenčního diagramu jsem znázornil základní postup při tvorbě projektu. Uživatel vytvoří nový projekt se zadaným názvem. Poté bude chtít do projektu přidat kritéria, stanoví počet kritérií a dojde k vytvoření objektu kritéria. Následně je umožněno vložení dalších dat o kritériích. Po úspěšné editaci kritérií proběhne naprosto stejným způsobem vytvoření a vložení variant. Jakmile projekt obsahuje všechna potřebná data kritérií i variant, je možné zadat výpočet výsledků. Celý proces je možné vidět na následujícím sekvenčním diagramu (Obr. 9).



Obr. 9: MULTI – Sekvenční diagram

4.1.3 Třídní diagram

Poslední nezbytnou částí objektové analýzy tvoří třídní diagram. Jeho úkolem je přehledně umožnit znázornění tříd, které budou použity ve vyvíjené aplikaci. Třídy si můžeme představit jako šablony, jež budou použity při tvorbě objektů. Objekty pak můžeme definovat jako seskupení dat a funkcionalit.



Obr. 10: MULTI – třídní diagram

Třídní diagram (Obr. 10) zobrazuje základní třídy celé aplikace. Třída Projekt je pro nás stěžejní jelikož obsahuje proměnné kriteria a varianty, které jsou objekty tříd Kriteria a Varianty. Diagram dále obsahuje také třídy reprezentující jednotlivé formuláře, třída MULTIView je hlavním formulářem celé aplikace, poté je zde uvedena třída Kriteria_prehled jako další formulář a naznačena je tu i třída Formular, která zastupuje ostatní potřebné formuláře. Jako poslední je zde třída IO, která zajišťuje vstupně výstupní funkce.

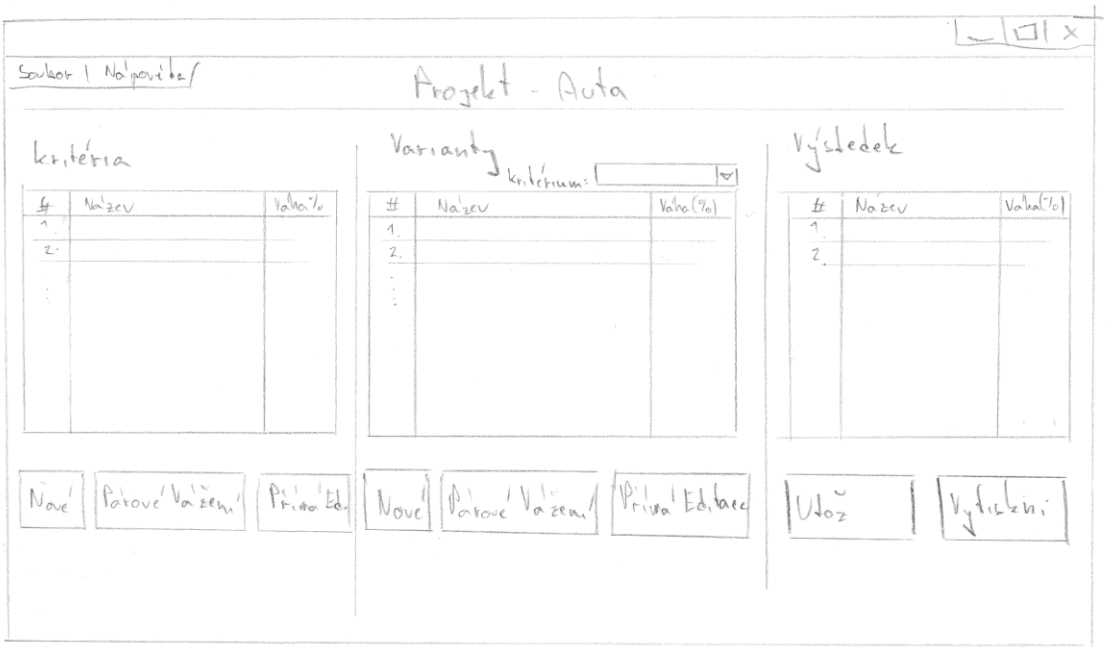
Jak je možné zjistit z diagramu, některé třídy obsahují základní funkce typu „set“ a „get“ pro vlastní proměnné, kterou jsou typu „private“ a proto jsou pro ostatní třídy nepřístupné. Třída Projekt navíc obsahuje sadu funkcí nezbytných pro realizaci všech základní výpočtů.

Třídy reprezentující formuláře obsahují vždy funkci „initComponents“, která vytvářejí všechny komponenty tvořící grafickou podobu daného formuláře. Ostatní funkce pak přísluší použitým tlačítkům či kontrolním funkcím.

4.2 Design aplikace

Design celé aplikace vychází z funkcionality původní aplikace IVMULTI a při jeho tvorbě byl kladen důraz na jednoduchost a účelnost. Při prvním spuštění aplikace může mít uživatel dojem, že se jedná o komplikovanou aplikaci, avšak záhy po začátku práce s aplikací zjistí, že je intuitivně veden k dalšímu kroku a postupně jsou mu zpřístupňovány další možnosti vytváření a editace.

Návrhy všech formulářů byly původně vytvořeny ručně jako nákresy a posléze implementovány ve vývojovém prostředí.



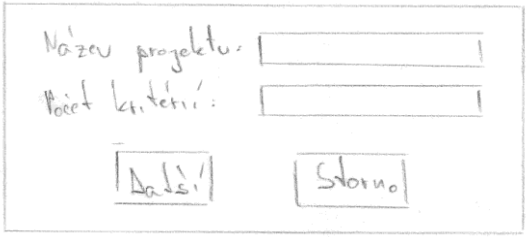
Obr. 11 – MULTI – hlavní obrazovka

Po spuštění aplikace se uživateli rovnou zobrazí hlavní obrazovka (Obr. 11), která obsahuje základní Menu a je rozčleněna na 4 základní části:

1. Název projektu
2. Kritéria
3. Varianty
4. Výsledek

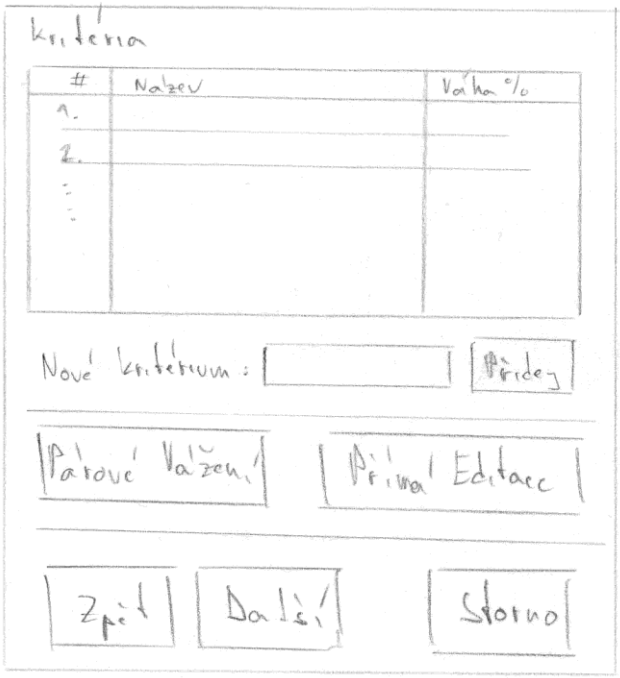
Cílem je zobrazit uživateli strukturu celé aplikace. Pro další kroky se zobrazí vždy příslušné speciální formuláře, které umožní pokračování v projektu. Logicky je celý proces rozdělen do dvou hlavních etap.

První z nich zahrnuje vytvoření projektu samotného a následně zabývá zadáním a specifikací kritérií:



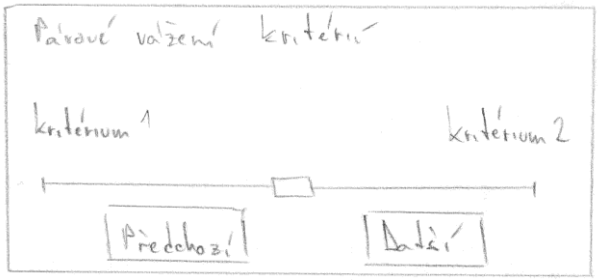
Obr. 12: MULTI – Nový projekt + počet kritérií

Jednoduchý formulář pro zadání názvu projektu a počtu kritérií (Obr. 12).



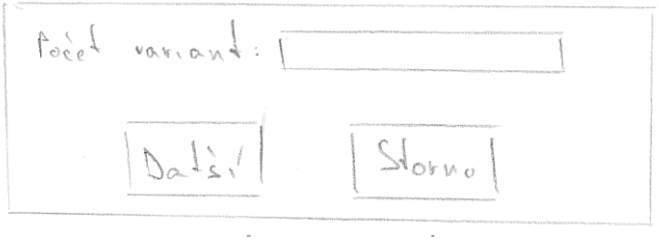
Obr. 13 – MULTI – Přehled kritérií

Tento formulář nazvaný Přehled kritérií (Obr. 13.) nám zobrazuje veškeré detaily kritérií. Zde dochází k přidání všech požadovaných kritérií. Umožňuje také jejich editaci ať už metodou párového vážení nebo je lze editovat přímo.



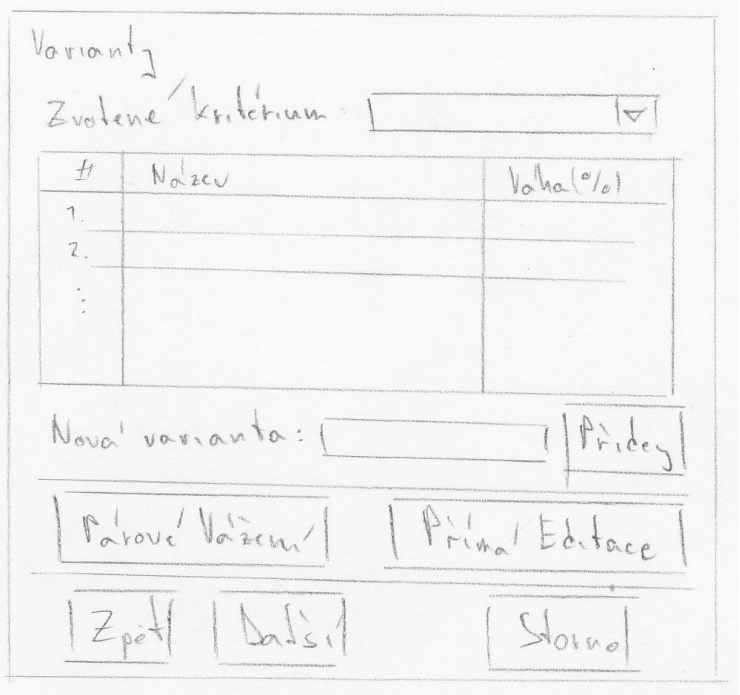
Obr. 14 – MULTI – Párové vážení kritérií

Párové vážení kritérií (Obr. 14) je zásadní formulář nezbytný pro uložení vah jednotlivých kritérií. Výpočet vah probíhá postupně porovnáváním vždy 2 kritérií mezi sebou a určením jejich důležitosti.



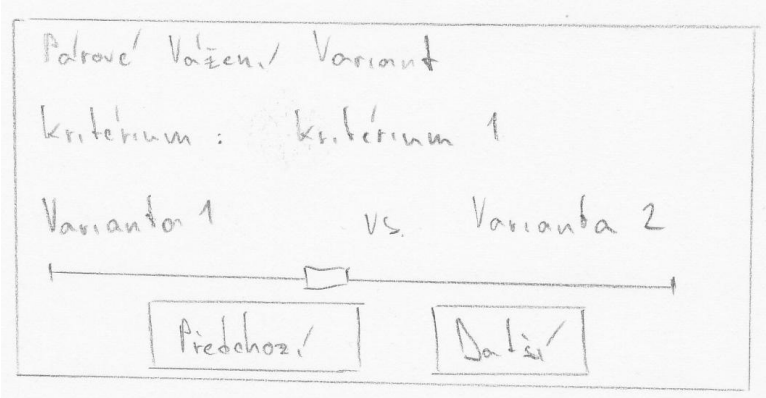
Obr. 15: MULTI – Počet variant

Formulář (Obr. 15) velmi podobný Počtu kritérií. Slouží pouze k zadání počtu variant.



Obr. 16 MULTI – Přehled variant

Pomocí tohoto formuláře (Obr. 16) je možné opět zadat všechny varianty a editovat je jak párovým vážením, tak přímo. Změna oproti Přehledu kritérií spočívá v možnosti výběru požadovaných vah variant podle dříve zadaných kritérií.



Obr. 17. MULTI – Párové vážení variant

Tento formulář (Obr. 17) opět realizuje párové vážení. V tomto případě se jedná o varianty.

Tyto grafické návrhy zahrnují základní nezbytné prvky pro práci s aplikací v plně grafickém prostředí. Součástí aplikace budou také další formuláře, které však budou mít pouze funkci oznamovací či varovnou a proto se není třeba zabývat jejich detailním zpracováním.

4.3 Programovací jazyk a implementační prostředí

Jako programovací jazyk jsem zvolil jazyk Java. Důvodem byla znalost tohoto jazyka a také především jeden z požadavků na aplikaci a to multiplatformnost. Java je totiž tzv. interpretovaný jazyk, což znamená možnost spuštění aplikace na jakémkoli počítači, který má nainstalovaný virtuální stroj jazyka Java.

Pro vývoj aplikace samotné jsem zvolil vývojové prostředí NetBeans v poslední verzi 6.8. Toto prostředí umožňuje jednoduché vytváření grafických aplikací, jejich kompilaci a ladění. Také umí zpracovat výsledný projekt do jednoduše spustitelného JAR souboru a podporuje také generování dokumentace.

4.4 Shrnutí a závěr

Díky realizované objektové analýze a grafickému návrhu jsem získal mnohem podrobnější představu o celé aplikaci. Je zřejmé, jaké části musí program obsahovat a kde tudíž začít. Bez těchto provedených analýz by se snadno mohlo stát, že by byly vytvořeny části, které by spolu nemusely být kompatibilní a mohly by tak způsobit nechtěné problémy.

5. Implementace

V následující kapitole se budu zabývat implementací celé aplikace. Budou zde vypsány nejdůležitější části zdrojového kódu a jejich vysvětlení. Také zde budou uvedeny hlavní změny oproti původní aplikaci IVMULTI.

5.1 Návrh struktury

Pro začátek na celé aplikaci jsem v prostředí NetBeans vytvořil nový projekt, nazvaný dle aplikace, MULTI. Java standardně pracuje s tzv. balíky (packages) a proto byl v rámci projektu vytvořen balík multi, do kterého byly také implicitně vytvořeny základní třídy pro aplikaci s grafickým rozhraním. Vzhledem k rozsahu celé aplikace jsem se rozhodl ponechat vše v jednom balíku. Třídy si však můžeme logicky rozdělit na část realizující výpočty a část vytvářející grafické rozhraní celé aplikace.

5.2 Implementace základní funkcionality

Hlavní třídou aplikace jako takové je automaticky vytvořená třída MULTIApp. Při následné implementaci jsem se však primárně zabýval třídou MULTIView, která byla také automaticky vytvořena v rámci nového projektu a jejíž hlavní úlohou je inicializace grafického rozhraní hlavní obrazovky aplikace. Pro budoucí postup však bylo nutno prvně implementovat základní třídy Projekt, Kriteria a Varianty, aby bylo možné definovat hlavní proměnné a postupně přidávat jednotlivé funkce. V případě tvorby složitější aplikace, která by mohla být rozdělena i mezi více osob, by určitě mělo smysl zvlášť realizovat práci na straně grafického rozhraní a zvlášť na funkční stránce. Jelikož jsem však aplikaci implementoval sám, rozhodl jsem se striktně nedodržovat toto rozdělení a postupně tak přibýval kód na obou stranách. Výhodou tohoto postupu byla možnost vždy si vyzkoušet, zda tento postup má smysl a jestli tak nevzniknou zbytečné překážky na některé straně.

5.2.1 Kriteria, Varianty, Projekt

V této části se budu věnovat třídám Kriteria, Varianty a Projekt. Jejich obsahem jsou proměnné obsahující všechna data a naprostá většina funkcí, které s nimi pracují. Do těchto tříd byla přepsána většina kódu z původní aplikace IVMULTI, které bylo možné zachovat a znovu využít.

5.2.1.1 Třída Kriteria

```
public class Kriteria {

    private int pocet_kriterii;
    private String[][] kriteria;
    int naplneni;

    public Kriteria(int pocet_kriterii) {
        this.pocet_kriterii = pocet_kriterii;
        this.kriteria = new String[pocet_kriterii][2];
        this.naplneni = 0;
}
```

Tato třída obsahuje 3 základní proměnné reprezentující počet zadaných kritérií, dvourozměrné pole názvů a vah jednotlivých kritérií a počet naplnění pole kritérií. První dvě proměnné jsou privátní a jsou přístupné pouze pro funkce obsažené v této třídě. Proto je nutné definovat veřejné funkce, které k nim budou moci přistupovat. Jako příklad uvedu funkce pro nastavení a vrácení počtu kritérií:

```
public void setPocet(int pocet_kriterii){
        this.pocet_kriterii = pocet_kriterii;
}
public int getPocet(){
        return this.pocet_kriterii;
}
```

Součástí této třídy jsou také další funkce zajišťující nastavení jednotlivých prvků pole kritérií, či pole celé.

5.2.1.2 Třída Varianty

```
public class Varianty {

    private int pocet_variant;
    private int pocet_kriterii;
    private String[][] varianty;
    int naplneni;

    public Varianty(int pocet_variant, int pocet_kriterii) {
        this.pocet_variant = pocet_variant;
        this.pocet_kriterii = pocet_kriterii;
        this.varianty =
                  new String[pocet_variant][pocet_kriterii+1];
        this.naplneni = 0;
}
```

Struktura této třídy je velmi podobná struktuře Kriterii. Pole variant však již nemá pevně zadaný druhý rozměr a proto je nutné jej také nastavit dynamicky. I tato třída obsahuje funkce typu „set“ a „get“ pro nastavení a získání hodnoty dané proměnné. Jsou napsány velmi podobně jako v předchozím příkladu a proto je zde nebudu opakovat.

5.2.1.3 Třída Projekt

Jedná se o stěžejní třídu celé aplikace. Umožňuje přístup jak k jednotlivým kritériím a variantám, tak především k jejich funkcím. Zajímavosti na této třídě je ten fakt, že byla implementována jako tzv. singleton, což má řadu výhod, které byly využity v rámci celé aplikace.

Ukázka zjednodušené implementace singletonu Projekt:

```
public class Projekt {

      private static Projekt instance;

         public Projekt(){
      }

         public static void ZalozProjekt() {
                  instance = new Projekt();
      }

         public static Projekt getInstance(){
                  if (instance == null) instance = new Projekt();
                  return instance;
      }
}
```

Princip singletonu je velmi prostý. V rámci běhu celé aplikace je vytvořena pouze jediná instance dané třídy, na kterou je možné se odkázat z kterékoli třídy a funkce. Vytvoření instance třídy Projekt a přístup k jejím funkcím je názorně vidět na následujících příkladech:

```
Projekt projekt = Projekt.ZalozProjekt();
```

Instance třídy projekt byla vytvořena a nyní k ní a jejím funkcím lze přistupovat odkudkoli pomocí tohoto kódu s využitím tečkové notace:

```
Projekt projekt = Projekt.getInstance();
projekt.setNazevProjektu(nazev_projektu);
```

Součástí této třídy jsou také proměnné kriteria a varianty reprezentující objekty tříd Kriteria a Varianty, které byly již detailněji popsány dříve.

Nejdůležitějších částí jsou však funkce samotné. Především se jedná o funkce realizující hlavní výpočty celé aplikace. Jako příklad zde uvedu funkci, jejíž náplní je spravovat uživatelské volby důležitostí jednotlivých kritérií mezi sebou na váhy příslušné pro každé kritérium.

```
public void parove_vazeni_kriterii(float[] volby) {
        float[][] mat_vah = new float[pocet_kriterii][pocet_kriterii];
        float[] vaha = new float[pocet_kriterii];
        int v = 0;

        for (int i = 0; i < pocet_kriterii; i++){
             mat_vah[i][i] = 1;
             for (int j = i+1; j < pocet_kriterii; j++)
                {
                 mat_vah[i][j] = volby[v];
                 mat_vah[j][i] = 1/mat_vah[i][j];
                 v= v+ 1;
                 }
        }

      vlastni_vektor(pocet_kriterii,(float)0.000001,mat_vah,vaha,'K');
        for (int i = 0; i < pocet_kriterii; i++){
            kriteria.kriteria[i][1] = String.valueOf(vaha[i]*100);
        }
       }
```

Tato funkce je volána s parametrem pole volby, které obsahuje všechna uživatelská hodnocení důležitostí při párovém vážení kritérií. Počet prvků tohoto pole je dán výpočtem kombinací bez opakování vždy pro 2 prvky z celkového počtu rovnu počtu kritérií. Pole volby je vytvořeno funkcí ve třídě Kriteria_parove_vazeni, která má za úkol provést uživatele ohodnocením všech kritérií a poté předat kompletní pole voleb.

Tato třída obsahuje dalších 7 funkcí, které byly převzaty z původní aplikace. Jejich účelem je spočítat a zkontrolovat váhu také pro varianty a ve finále vypočítat výsledek celého multikriteriálního hodnocení variant.

5.2.2 Export a import

Export dat projektu probíhá do souboru XML, ze kterého je pak také možné daný projekt načíst. Pro obě vstupně výstupní operace se využívá třídy IO, která je opět deklarována jako singleton. Vygenerovaný XML soubor je sice čitelný, avšak jeho použití je primárně určeno pouze pro tuto aplikaci. Pro ukázku zde uvedu jeho část.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<multi>
    <nazevProjektu>Auta</nazevProjektu>
    <pocetKriterii>3</pocetKriterii>
    <pocetVariant>3</pocetVariant>
    <kriteria>
        <item>Cena auta</item>
        <item>29.658216</item>
    </kriteria>
    …
</multi>
```

5.3 Kompletace grafické a funkční části

Jak jsem psal již při implementaci základní funkcionality, pracoval jsem zároveň i na grafickém rozhraní. Proto bylo možné během vývoje postupně provádět první testy vytvořené funkcionality. Všechny formuláře byly z velké části implementovány přesně podle grafického návrhu. Místy došlo k drobným změnám, protože plánovaná funkcionalita se nakonec ukázala jako zbytečná.

K původním navrženým formulářům byly také doimplementovány podpůrné formuláře pro možnosti ukládání a načítání souborů, či informativní, varovné nebo rozhodovací dialogy.

Jakmile byla kompletně dokončena základní funkcionalita, bylo možné pokročit k další fázi a to testování, opravy chyb a napsání uživatelské příručky.

6. Testování a uvedení do provozu

6.1 Testování

Hlavní testování probíhalo přímo na počítači, kde byla aplikace vyvíjena. Jedná se počítač se systémem Windows 7, CPU Intel C2D 1,80Ghz, 3GB RAM. V prostředí NetBeans jsem hojně využil tzv. DEBUG mód, který umožňuje zastavit aplikaci v konkrétním bodě a poté jednotlivé další řádky zdrojového kódu krokovat. Je možné se poté dostat do jednotlivých funkcí a zjistit

k jakým změnám dochází např. během cyklů nebo zda náhodou nedošlo k přeplnění některých proměnných.

Pro testování jsem také využil další počítače různých konfigurací, aby bylo možné zaručit spustitelnost aplikace na kterémkoli stroji s nainstalovaným virtuální strojem Java.

6.2 Uživatelská příručka

Aby bylo možné využívat plnou kapacitu aplikace, je nutné vytvořit uživatelskou příručku, která provede uživatele krok za krokem při cestě od zadání vstupních dat až po získání výsledku. Na následujících stránkách bude zobrazena tato uživatelská dokumentace, která nám zároveň bude demonstrovat plnou funkčnost vyvíjené aplikace.

Aplikace MULTI – Uživatelská příručka

O aplikaci:

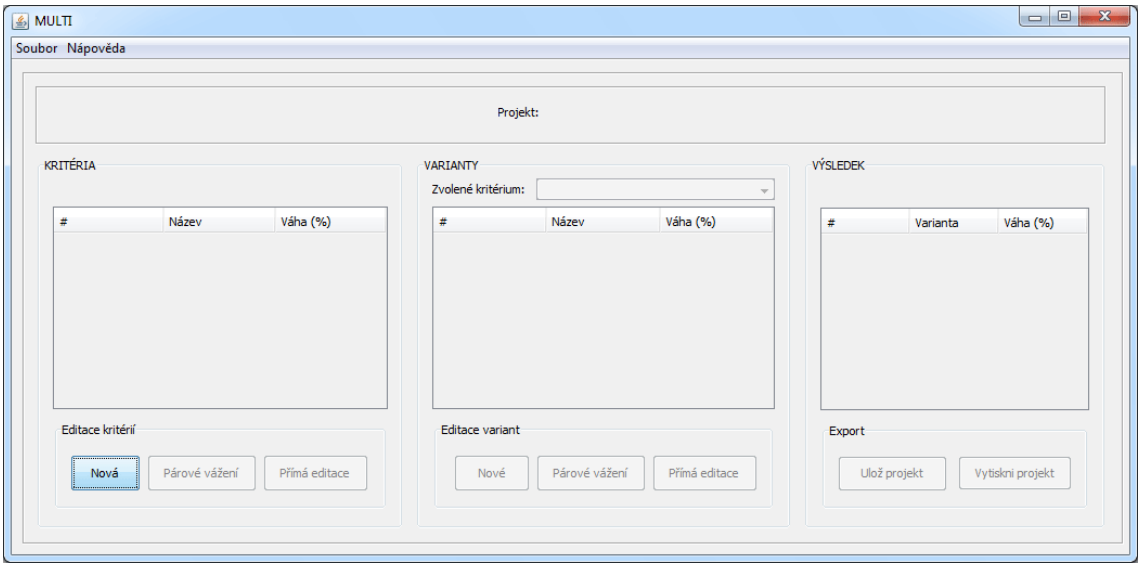
Program MULTI umožňuje využít matematické metody multikriteriálního hodnocení variant pro usnadnění rozhodnutí při výběru mezi více možnostmi (variantami), u nichž si nejsme jistí, které charakteristiky (kritéria) jsou pro nás více důležité. Aplikace umožní zadat nejdůležitější kritéria výběru, specifikovat jejich důležitost a následně zadat varianty, jež chceme zhodnotit. Program nás provede postupných ohodnocením důležitosti všech variant v rámci zadaných kritérií a následně zobrazí výsledek. Metodu celého výpočtu můžeme najít sepsanou v Uživatelském manuálu programu IVMULTI, který je předchůdcem této aplikace.

Práce s aplikací:

Kompletní aplikace se nachází v adresáři Multi. Aby mohla být úspěšně spuštěna, je nutné neměnit obsah tohoto adresáře a je také nezbytné mít nainstalovaný jazyk Java, konkrétně Java Runtime Enviroment (JRE), který lze najít na stránkách http://www.java.com.

Funkcionalitu celé aplikace si nyní budeme demonstrovat na vzorovém příkladu výběru auta, který představuje složitější rozhodnutí a zároveň nemalou investici. Vybíráme mezi 3 typy (variantami) aut: Škoda Fabia, Fiat Punto a Volkswagen Golf a pro náš výběr jsou rozhodující 3 kritéria: Cena auta, Provozní náklady a Spolehlivost. Program MULTI nám umožní postupně určit co je pro nás nejdůležitější a vypočte nám nejlepší variantu dle zadaných parametrů.

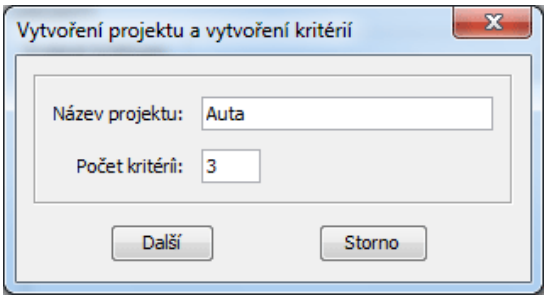
Pro úspěšné spuštění aplikace spusťte soubor MULTI.jar a zobrazí se následující obrazovka:



Úvodní obrazovka

Tato obrazovka představuje hlavní okno programu. Na začátku je celé prázdné, jelikož nebyly zadány žádné hodnoty. Naším dalším cílem je založit nový projekt, který nám umožní určit pro nás nejvýhodnější auto.

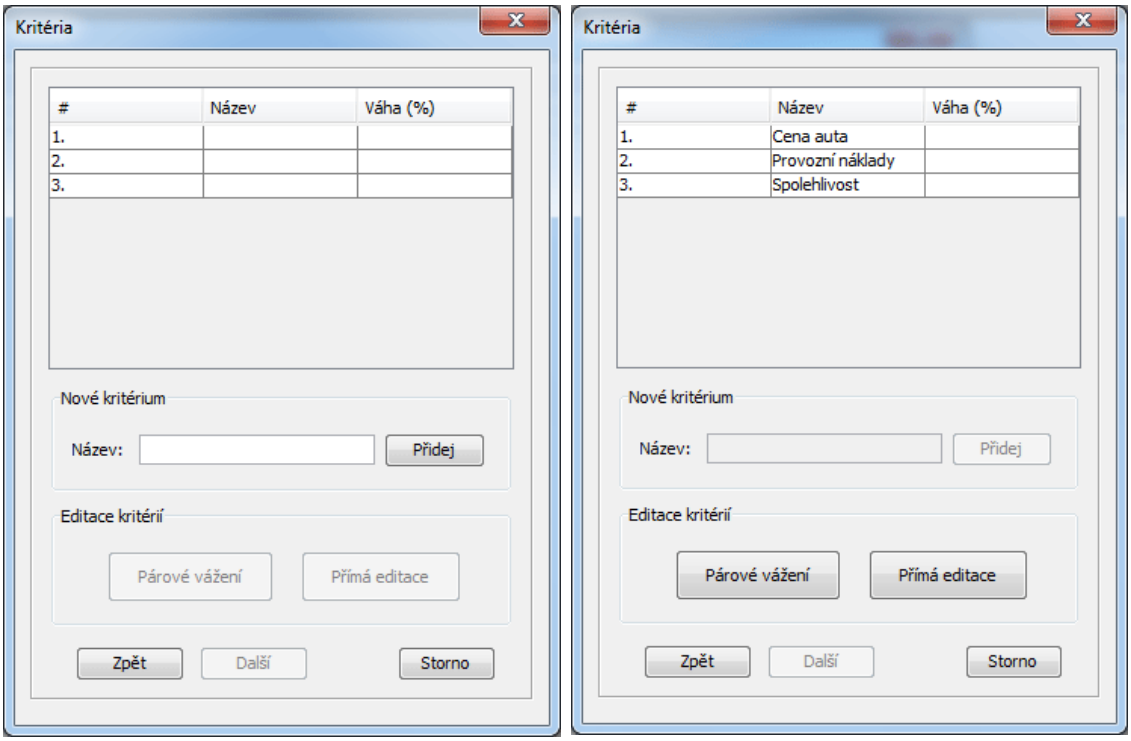
Nový projekt lze spustit z Menu položkou Nový projekt, nebo lze kliknout na tlačítko Nová v levém dolním okraji formuláře. Zobrazí se následující dialog:



Vytvoření projektu a kritérií

Zde je nutné zadat Název projektu a Počet kritérií. My jsme projekt logicky nazvali Auta a počet kritérií zvolili 3 přesně podle našich požadavků. Počet kritérií není omezen, ovšem z praktického hlediska je vhodné uvažovat maximálně okolo 10 kritérií.

Pro pokračování použijte tlačítko Další:

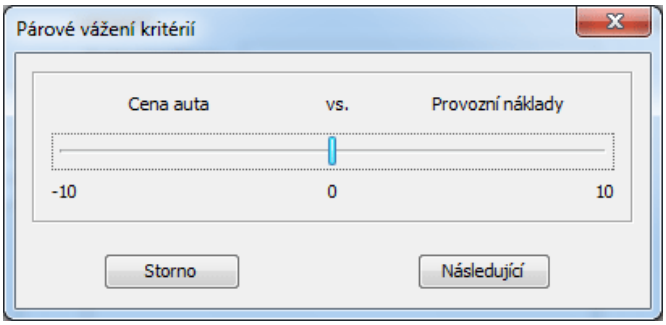


Kritéria před a po zadání názvů

Následující dialog je zobrazen dvakrát, protože je zde ukázáno chování aplikace při dalším postupu. Po zadání kritérií se zobrazí prázdný formulář a našim úkolem je zadat názvy vybraných kritérií. Dokud je všechny nezadáme, aplikace nás nepustí dále. My jsme opět zadali námi zvolená 3 kritéria a otevřel se nám prostor pro pokračování.

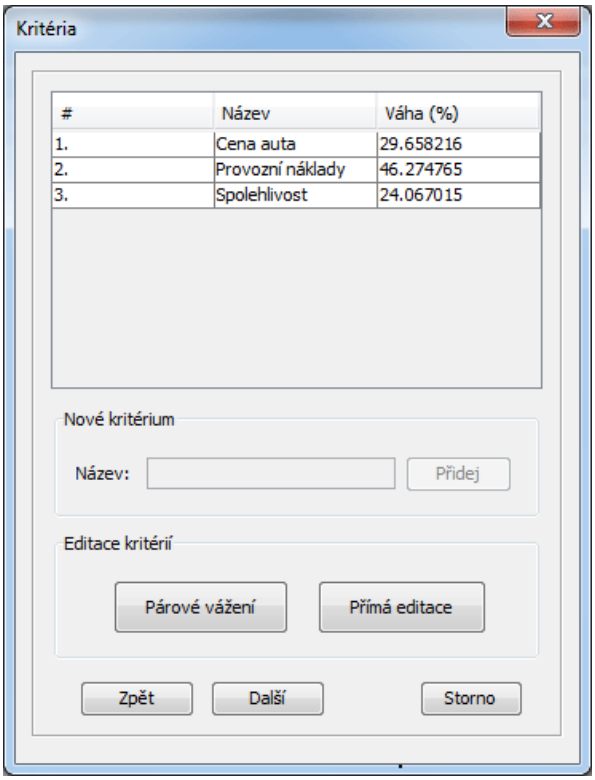
Nyní máme na výběr ze dvou možností: Párové vážení a Přímou editaci. My zvolíme Párové vážení a k přímé editaci se později ještě vrátíme.

Otevře se následující dialog:



Párové vážení kritérií

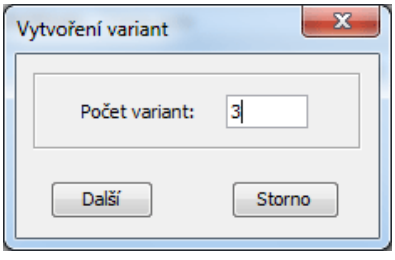
Teď musíme určit, která kritéria jsou pro nás důležitější. Aplikace nám zobrazí vždy kombinaci právě dvou zvolených kritérií. Díky posuvníku můžeme určit, zda daná kritéria jsou stejně důležitá (jako v našem případě) nebo zda je důležitější kritérium první či druhé. Na stupnici poté odhadneme jak moc důležitější pro nás je dané kritérium. Je nutné takto projít všechny kombinace zadaných kritérií. Pokračování zvolíte tlačítkem Následující:



Zadaná kritéria i váhy

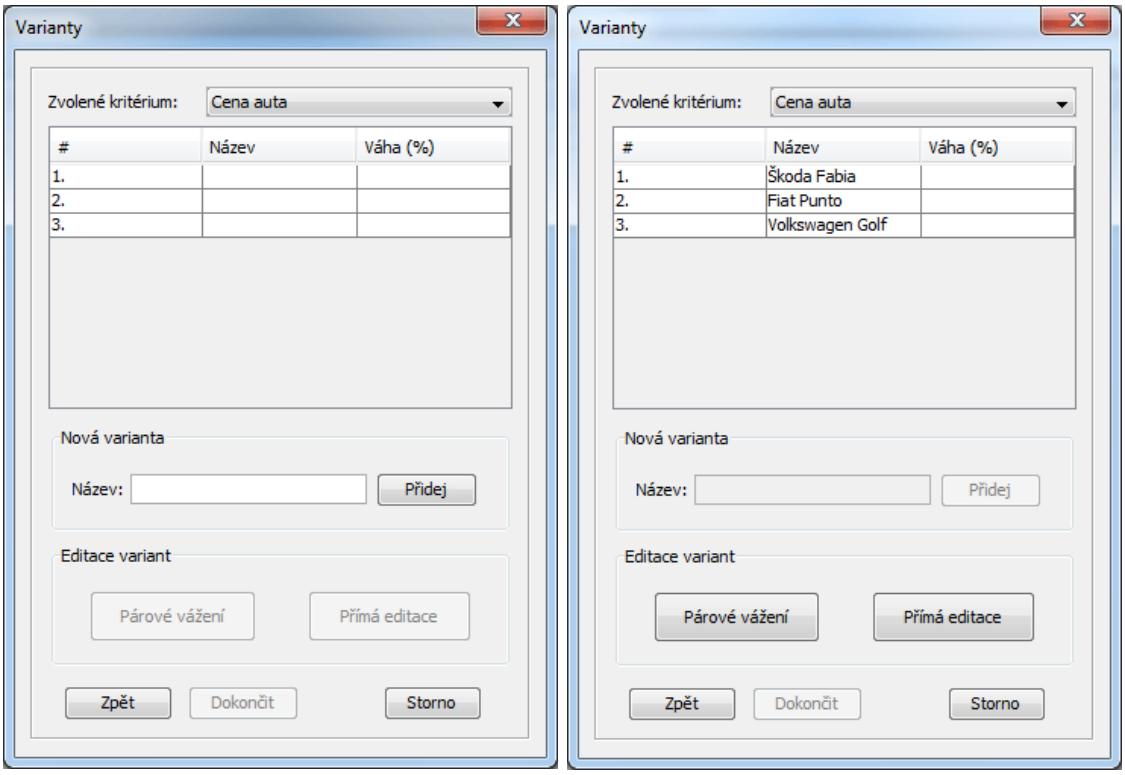
Jakmile porovnáme všechna kritéria, aplikace nám automaticky přepočítá naše volby na váhy jednotlivých kritérií mezi sebou. Na našem příkladu je možné vidět, že nejdůležitějším kritériem pro volbu auta jsou Provozní náklady.

Nyní můžeme pokračovat dále, ovšem pokud jsme zadali některou hodnotu špatně, či chceme změnit naši volbu, je možné kritéria přímo editovat pomocí tlačítka Přímá editace. Jeho stisknutím se nám zpřístupní tabulka zadaných hodnot, jež můžeme editovat. Editaci ukončíme opětovným stisknutím tlačítka. Po úspěšném natavení všech kritérií můžeme pokračovat tlačítkem Dále.



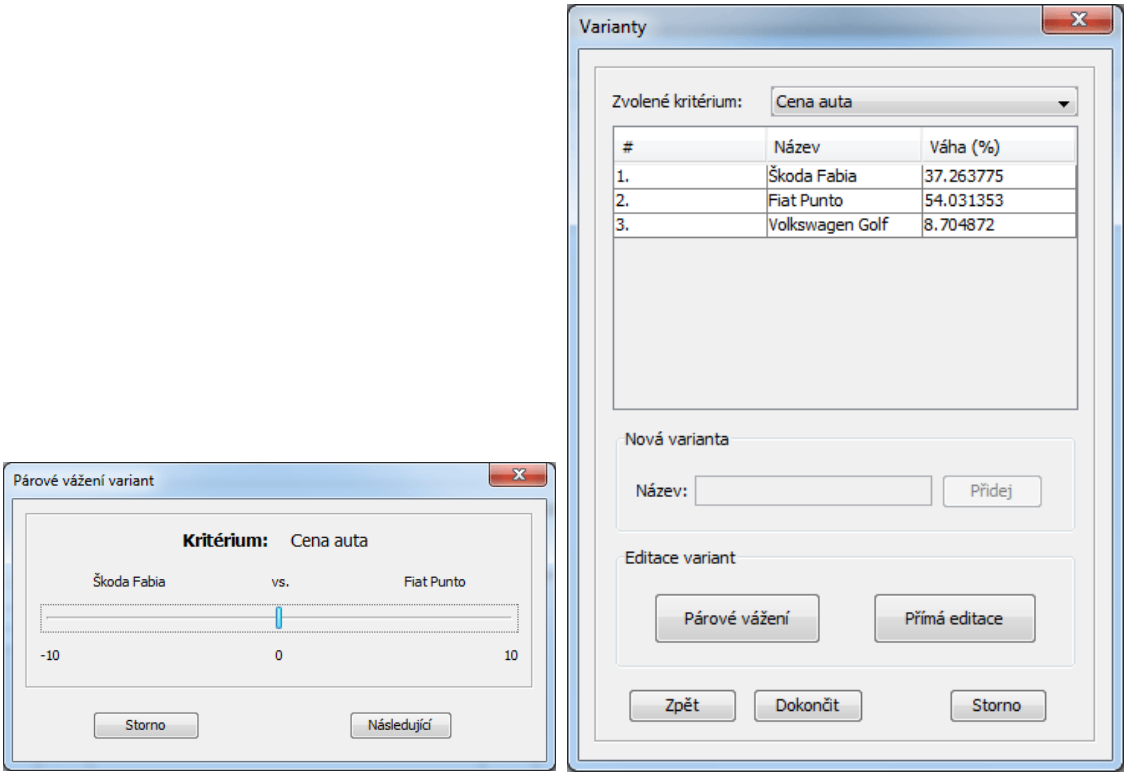
Vytvoření variant

V následujícím dialogu pouze zadáme počet variant. Jejich množství opět není omezeno, ovšem reálně se většinou vejdeme do 10 variant. Pokračovat můžeme tlačítkem Další.



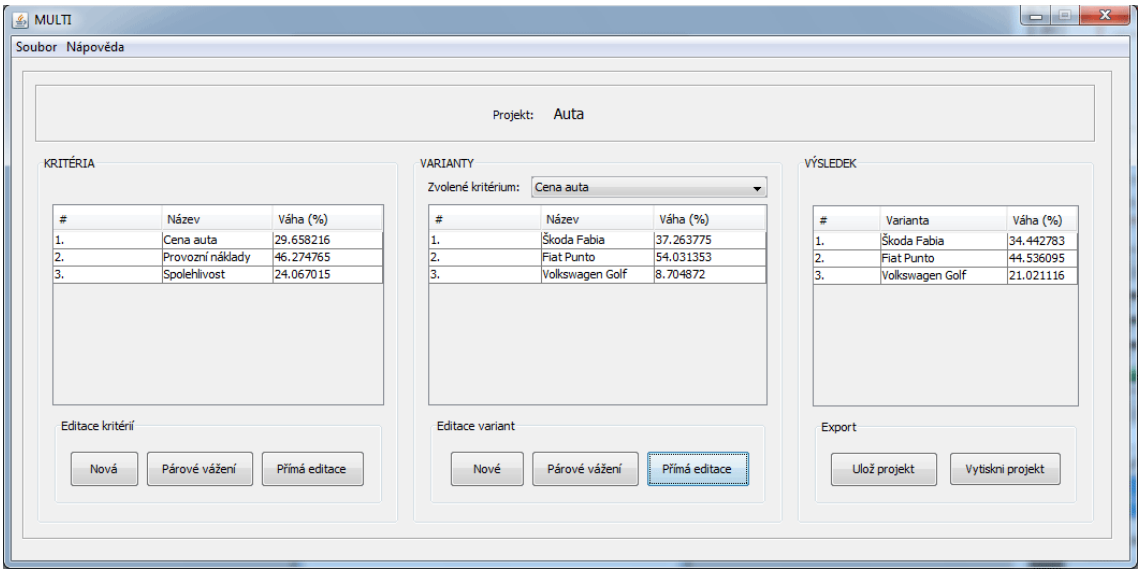
Varianty před a po zadání názvů

Princip zadání názvů variant je naprosto stejný jako u zadání kritérií. Na formuláři přibyla možnost ukázat váhy variant pro jednotlivá kritéria, ovšem bez zadaných dat zde není co zobrazit. Můžeme proto opět pokračovat Párovým vážením nebo Přímo editovat.



Párové vážení variant, Zadané varianty a váhy

Párové vážení opět probíhá obdobně jako u variant jen s tím rozdílem, že je vždy uvedeno kritérium, pro které dané varianty hodnotíme. V našem případě Rozhodujeme o ceně mezi Škodou Fabií a Fiatem Punto. Naše volba značí, že jejich ceny vnímáme jako přibližně stejné. Po zvážení všech variant se nám naše volby opět přepočítají a vypíšou. I nyní můžeme zvolit Přímou editace, pokud chceme některou volbu změnit. V opačném případě můžeme pokračovat do finále a tlačítkem Dokončit si nechat vypočítat a zobrazit výsledky.



Výsledky

Na hlavní obrazovce nyní můžeme vidět všechny vypočtené váhy jednotlivých kritérií a variant. V posledním panelu pak nalezneme finální výsledek. Dle našeho zadání a ovážení nám nejlépe vyšlo auto Fiat Punto, následováno Škodou Fabií a poslední skončil Volkswagen Golf.

Hotový projekt je možné uložit do souboru XML a později se k němu kdykoli vrátit. Umožněn je také tisk, který zobrazí všechny zadané hodnoty kritérií, variant a výsledků.

Celé toto multikriteriální hodnocení variant je čistě subjektivní záležitostí. Získáme sice ohodnocení pořadí jednotlivých variant, ovšem to nám nezaručuje danou volbu jako nejlepší. Může nás však k tomu velmi přiblížit a proto přeji všem uživatelům této aplikace hodně úspěšných rozhodnutí.

7. Závěr

Cílem této bakalářské práce bylo vytvořit moderní aplikaci s grafickým rozhraním, využívající funkcionalitu původní aplikace IVMULTI a umožňující export dat do XML souborů. Postupně byly provedeny analýzy jak původního programu, tak nově vytvářené aplikace a výsledky obou analýz poté byly zpracovány a využity při následné implementaci. Nechybělo ani závěrečné otestování vytvořené aplikace a vytvoření uživatelské příručky pro práci s danou aplikací.

Myslím si, že aplikace MULTI splňuje zadané požadavky a umožňuje efektivně zpracovat zadaná kritéria a varianty na výsledek, který umožní jednodušší rozhodování v náročnějších případech.

Využití metody multikriteriálního hodnocení variant má rozhodně své místo v praktickém životě a já osobně plánuji využít tuto aplikaci na maximum. Až dlouhodobější zkušeností a praxí mohu zjistit nejčastější způsoby využití a podílet se tak na jejím dalším vývoji a optimalizaci právě na tyto konkrétní případy.

Během vývoje této práce jsem také zjistil jaké výhody a nevýhody přináší implementace obdovné aplikace v jazyce Java a prostředí NetBeans. Tyto zkušenosti rozhodně využiji v praxi ať už při rozvoji programu samotného MULTI či při implementaci jakékoli jiné aplikace.

Použitá literatura:

1. Vondrák, I.: Uživatelský manuál systému IVMULTI – Multikriteriální hodnocení variant
2. Pecinovský, R.: Myslíme objektově v jazyku Java, 2. aktualizované vydání, Grada Publishing a.s., 2009
3. Kanisová, H., Müller, M.: UML srozumitelně, 2. aktualizované vydání, Computer Press, a.s, 2007
4. Herout, P.: Java - grafické uživatelské prostředí a čeština, KOPP, 2004

Přílohy:

1. Vondrák, I.: Uživatelský manuál systému IVMULTI – Multikriteriální hodnocení variant